最优化理论大作业

题目描述

给定一个定义在n维空间中的凸函数 $f(\mathbf{x})$ ，在空间中任取m个点 $\{x_i\}_{i=1}^m$,另记 $p_i = \frac{1}{2}\nabla f(x_i), i = 1, \cdots, m$

再为每个点定义一个权重 $\omega_i, i = 1, \cdots, m, \forall\omega_i$ 满足如下约束：

$$\forall j \neq i, ||\mathbf{x}_i - \mathbf{p}_i||^2 - w_i \leq ||\mathbf{x}_i - \mathbf{p}_j||^2 - w_j, j = 1, \cdots, m$$

已知 $f(\mathbf{x}), \{\mathbf{x}_i\}_{i=1}^m, \{\mathbf{p}_i\}_{i=1}^m$ ，求m维线性规划问题 $\min \sum_{i=1}^m w_i$

问题分析

要求解该线性规划问题，其基本形式为：

$$\text{s.t.} \begin{Bmatrix} \min \mathbf{c}^T\omega \\ \forall j \neq i, ||\mathbf{x}_i - \mathbf{p}_i||^2 - w_i \leq ||\mathbf{x}_i - \mathbf{p}_j||^2 - w_j, j = 1, \cdots, m \\ A\omega \leq \mathbf{b} \end{Bmatrix}$$

其中将约束条件 $\forall j \neq i, ||\mathbf{x}_i - \mathbf{p}_i||^2 - w_i \leq ||\mathbf{x}_i - \mathbf{p}_j||^2 - w_j, j = 1, \cdots, m$ 进行转化。

$$w_j - w_i \leq ||\boldsymbol{x}_i - \boldsymbol{p}_j||_2 - ||\boldsymbol{x}_i - \boldsymbol{p}_i||_2, \forall j \neq i, j \in \{1, \ldots, m\}$$

即

$$\begin{aligned} & w_1 - w_i \leq ||\boldsymbol{x}_i - \boldsymbol{p}_1||_2 - ||\boldsymbol{x}_i - \boldsymbol{p}_i||_2 \\ & w_2 - w_i \leq ||\boldsymbol{x}_i - \boldsymbol{p}_2||_2 - ||\boldsymbol{x}_i - \boldsymbol{p}_i||_2 \\ & \cdots\cdots \\ & w_m - w_i \leq ||\boldsymbol{x}_i - \boldsymbol{p}_m||_2 - ||\boldsymbol{x}_i - \boldsymbol{p}_i||_2 \end{aligned}$$

其中 $j \neq i$

联加得到：

$$\sum_{j=1, j\neq i}^m w_j - (m-1)w_i \leq \sum_{j=1, j\neq i}^m ||\boldsymbol{x}_i - \boldsymbol{p}_j||_2 - (m-1)||\boldsymbol{x}_i - \boldsymbol{p}_i||_2$$

即

$$\sum_{j=1}^m w_j - mw_i \leq \sum_{j=1}^m ||\boldsymbol{x}_i - \boldsymbol{p}_j||_2 - m||\boldsymbol{x}_i - \boldsymbol{p}_i||_2$$

然后根据该约束条件可知A和b：

$$A\omega = \sum_{j=1}^m w_j - mw_i$$

$$\boldsymbol{A} = \begin{pmatrix} 1-m & 1 & \ldots & 1 \\ 1 & 1-m & \ldots & 1 \\ 1 & \cdots & 1-m & 1 \\ 1 & \cdots & 1 & 1-m \end{pmatrix}$$

$$\boldsymbol{b} = (b_1, b_2, \ldots, b_m)^{\mathrm{T}}$$

$$b_i = \sum_{j=1}^m ||\boldsymbol{x}_i - \boldsymbol{p}_j||_2 - m||\boldsymbol{x}_i - \boldsymbol{p}_i||_2$$

从而有 $A\omega \leq \mathbf{b}$

解题过程

- 示例1

$$f(\mathbf{x}) = \mathbf{x}^T M\mathbf{x}, x = (x_0, x_1)^T \quad , M = \begin{pmatrix} 5 & 0 \\ 0 & 6 \end{pmatrix}$$

即 $f\left(\mathbf{x} = (x_0, x_1)^T\right) = 5x_0^2 + 6x_1^2$

给定凸函数的梯度 (解析解)： $\nabla f(\mathbf{x}) = (10x_0, 12x_1)$

$$\mathbf{p}_i = \frac{1}{2}\nabla f(\mathbf{x}_i), i = 1, \cdots, m$$

任意取m=50个点（显示精度为三位小数）

$$\{\mathbf{x}_i^T\}_{i=1}^m = \{(0.458, 0.825), (0.373, 0.333), (0.111, 0.223), (0.574, 0.095), (0.529, 0.35), (0.495, 0.277), (0.552, 0.301), (0.115, 0.637), (0.876, 0.524), (0.071, 0.403), (0.405, 0.044), (0.692, 0.009), (0.072, 0.702), (0.696, 0.599), (0.191, 0.551), (0.636, 0.405), (0.907, 0.229), (0.954, 0.594), (0.146, 0.722), (0.82, 0.279), (0.352, 0.219), (0.594, 0.614), (0.948, 0.196), (0.557, 0.025), (0.557, 0.531), (0.259, 0.522), (0.86, 0.855), (0.222, 0.178), (0.172, 0.329), (0.265, 0.742), (0.749, 0.224), (0.769, 0.949), (0.663, 0.076), (0.567, 0.321), (0.746, 0.394), (0.904, 0.22), (0.482, 0.999), (0.745, 0.298), (0.288, 0.933), (0.214, 0.652), (0.97, 0.314), (0.716, 0.639), (0.925, 0.219), (0.215, 0.768), (0.596, 0.154), (0.741, 0.75), (0.519, 0.876), (0.047, 0.478), (0.226, 0.352), (0.602, 0.001)\}$$

$$\{\mathbf{p}_i\}_{i=1}^m = \{(2.289, 4.951), (1.866, 1.999), (0.556, 1.338), (2.869, 0.568), (2.646, 2.1), (2.474, 1.662), (2.761, 1.807), (0.577, 3.825), (4.38, 3.145), (0.357, 2.419), (2.023, 0.267), (3.462, 0.057), (0.362, 4.21), (3.481, 3.594), (0.956, 3.304), (3.179, 2.428), (4.533, 1.375), (4.771, 3.562), (0.73, 4.334), (4.099, 1.676), (1.758, 1.316), (2.97, 3.684), (4.738, 1.175), (2.785, 0.153), (2.785, 3.186), (1.293, 3.13), (4.301, 5.128), (1.111, 1.067), (0.861, 1.975), (1.327, 4.449), (3.745, 1.343), (3.843, 5.692), (3.315, 0.454), (2.836, 1.928), (3.73, 2.365), (4.519, 1.317), (2.409, 5.996), (3.726, 1.785), (1.438, 5.595), (1.071, 3.91), (4.851, 1.887), (3.58, 3.835), (4.625, 1.315), (1.076, 4.607), (2.981, 0.925), (3.704, 4.5), (2.594, 5.255), (0.233, 2.866), (1.13, 2.113), (3.01, 0.008)\}$$

将A和计算得到的b传入scipy库的optimize.linprog函数中

得到

$$\omega = (2.37753034, 0., 0., 0., 0.37911533, 0., 0.28873194, 0.75304941, 2.31531278, 0., 0., 0.27156483, 1.05629031, 1.97322445, 0.39360804, 0.97922313, 1.56749607, 2.85911597, 1.25739811, 1.31659753, 0., 1.71268462, 1.67825632, 0., 1.22544728, 0.37072653, 3.6109386, 0., 0., 1.54737954, 0.87540685, 3.77475945, 0.20911997, 0.41757779, 1.36269631, 1.53482551, 3.3238196, 1.05512623, 2.60645015, 0.9770189, 2.0419519, 2.21028841, 1.62588109, 1.6052069, 0.05184735, 2.77645393, 2.77563597, 0., 0., 0.)$$

代码如下

```python
import numpy as np
from scipy import optimize

def fun(x):
    '''
    给定凸函数：y = 5 * x[0] * x[0] + 6 * x[1] * x[1]
    '''
    y = np.square(x)
    y[0] *= 5
```

```python
    y[1] *= 6
    return np.sum(y)

def grad_fun(x):
    '''
    给定凸函数的梯度（解析解）：grad(x) = (10 * x[0], 12 * x[1])
    '''
    p=[]
    for i in range(m):
        y =[]
        y.append(10 * x[i][0])
        y.append(12 * x[i][1])
        p.append(y)
    return np.array(p)
'''任取m个点
'''
m = 50
X = np.random.random(size=(m, 2))
P = 0.5*grad_fun(X)
print(np.round(X,3).tolist())
print(np.round(P,3).tolist())

c = np.ones((m, ))
A_ub = np.ones((m, m)) - np.identity(m) * m

'''求解b_ub
'''
t_ub = []
for i in range(m):
    t = np.sqrt(np.sum(np.square(X[i] - P), axis=-1))
    t[i] *= (1 - m)
    t_ub.append(np.sum(t))
b_ub = np.array(t_ub)
# 求解
res = optimize.linprog(c, A_ub=A_ub, b_ub=b_ub, bounds=[0, None], method='re
print(res)
```

LP.py

Users > jianglichai > study > homework > 最优化理论 > LP.py > ...

```python
'''求解b_ub
'''
t_ub = []
for i in range(m):
    t = np.sqrt(np.sum(np.square(X[i] - P), axis=-1))
    t[i] *= (1 - m)
    t_ub.append(np.sum(t))
b_ub = np.array(t_ub)
# 求解
res = optimize.linprog(c, A_ub=A_ub, b_ub=b_ub, bounds=[0, None], method='revised simplex')
print(res)
```

问题　输出　调试控制台　终端　JUPYTER

```
     con: array([], dtype=float64)
     fun: 57.15775739100661
 message: 'Optimization terminated successfully.'
     nit: 37
   slack: array([-7.10542736e-14,  1.18199334e+01,  7.54134500e+01,  8.15956603e+00,
        0.00000000e+00,  9.47427880e-01,  7.10542736e-15,  1.42108547e-14,
       -2.84217094e-14,  2.96256291e+01,  5.02488901e+01,  7.10542736e-15,
        4.44089210e-15, -2.13162821e-14,  0.00000000e+00, -7.10542736e-15,
        0.00000000e+00,  1.42108547e-14, -1.06581410e-14, -7.10542736e-15,
        3.87156006e+01, -7.10542736e-15,  1.42108547e-14,  1.65599350e+01,
        1.06581410e-14,  0.00000000e+00,  5.68434189e-14,  7.03452209e+01,
        4.10244909e+01,  1.42108547e-14,  1.77635684e-15, -1.70530257e-13,
        1.42108547e-14,  7.10542736e-15,  1.06581410e-14,  4.26325641e-14,
       -2.84217094e-14, -1.77635684e-15, -2.84217094e-14,  7.10542736e-15,
        4.97379915e-14, -2.84217094e-14, -6.03961325e-14,  0.00000000e+00,
        7.10542736e-15,  0.00000000e+00,  0.00000000e+00,  1.03384725e+01,
        2.88311587e+01,  7.31878446e+00])
  status: 0
 success: True
       x: array([2.37753034, 0.        , 0.        , 0.        , 0.37911533,
       0.        , 0.28873194, 0.75304941, 2.31531278, 0.        ,
       0.        , 0.27156483, 1.05629031, 1.97322445, 0.39360804,
       0.97922313, 1.56749607, 2.85911597, 1.25739811, 1.31659753,
       0.        , 1.71268462, 1.67825632, 0.        , 1.22544728,
       0.37072653, 3.6109386 , 0.        , 0.        , 1.54737954,
       0.87540685, 3.77475945, 0.20911997, 0.41757779, 1.36269631,
       1.53482551, 3.3238196 , 1.05512623, 2.60645015, 0.9770189 ,
       2.0419519 , 2.21028841, 1.62588109, 1.6052069 , 0.05184735,
       2.77645393, 2.77563597, 0.        , 0.        , 0.        ])
```

- 示例2

$$f(\mathbf{x}) = \mathbf{a}^T\mathbf{x}, x = (x_0, x_1)^T \quad , a = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

$$f\left(\mathbf{x} = (x_0, x_1)^T\right) = 5x_0 + 6x_1$$

给定凸函数的梯度 (解析解): $\nabla f(\mathbf{x}) = (5, 6)$

$$\mathbf{p}_i = \frac{1}{2}\nabla f(\mathbf{x}_i), i = 1, \cdots, m$$

任意取m=50个点（显示精度为三位小数）

$$\{\mathbf{x}_i^T\}_{i=1}^m = \{(0.964, 0.751), (0.928, 0.88), (0.6, 0.604), (0.957, 0.482), (0.278, 0.17), (0.501, 0.121), (0.238, 0.252), (0.669, 0.985), (0.842, 0.261), (0.626, 0.31), (0.796, 0.043), (0.09, 0.924), (0.824, 0.213), (0.364, 0.94), (0.759, 0.625), (0.398, 0.768), (0.227, 0.958), (0.396, 0.643), (0.36, 0.077), (0.458, 0.816), (0.765, 0.062), (0.654, 0.737), (0.945, 0.504), (0.296, 0.31), (0.846, 0.187), (0.159, 0.896), (0.99, 0.39), (0.737, 0.286), (0.056, 0.962), (0.437, 0.702), (0.321, 0.156), (0.422, 0.578), (0.84, 0.741), (0.594, 0.743), (0.962, 0.67), (0.08, 0.43), (0.087, 0.178), (0.195, 0.821), (0.15, 0.41), (0.512, 0.821), (0.905, 0.885), (0.189, 0.972), (0.568, 0.624), (0.58, 0.557), (0.318, 0.837), (0.611, 0.406), (0.9, 0.799), (0.871, 0.695), (0.581, 0.015), (0.387, 0.149)\}$$

$$\{\mathbf{p}_i\}_{i=1}^m = \{(2.5, 3.0), (2.5, 3.0)\}$$

将A和计算得到的b传入scipy库的optimize.linprog函数中

得到

$$
\begin{aligned}
\omega = (&2.07833750e-15, 1.22568622e-15, 2.43360887e-15, 6.75015599e-16, \\
&1.70530257e-15, 9.59232693e-16, 1.58095759e-15, 1.63424829e-15, \\
&6.75015599e-16, 8.70414851e-16, 2.66453526e-15, 2.87769808e-15, \\
&2.06057393e-15, 2.96651592e-15, 1.93622895e-15, 0.00000000e+00, \\
&1.11910481e-15, 2.25597319e-15, 9.23705556e-16, 2.00728323e-15, \\
&2.66453526e-15, 1.08357767e-15, 1.10134124e-15, 6.57252031e-16, \\
&1.45661261e-15, 1.63424829e-15, 1.10134124e-15, 2.13162821e-16, \\
&1.49213975e-15, 1.42108547e-15, 1.70530257e-15, 1.49213975e-15, \\
&1.68753900e-15, 9.94759830e-16, 1.38555833e-15, 1.66977543e-15, \\
&1.35003120e-15, 1.22568622e-15, 1.24344979e-15, 1.63424829e-15, \\
&1.24344979e-15, 1.82964754e-15, 9.94759830e-16, 0.00000000e+00, \\
&1.95399252e-15, 0.00000000e+00, 9.23705556e-16, 1.43884904e-15, \\
&2.25597319e-15, 1.43884904e-15)
\end{aligned}
$$

代码如下

```python
import numpy as np
from scipy import optimize

def fun(x):
    '''
    给定凸函数：y = 5 * x[0] + 6 * x[1]
    '''
    y = x
    y[0] *= 5
    y[1] *= 6
    return np.sum(y)

def grad_fun(x):
    '''
    给定凸函数的梯度（解析解）：grad(x) = (5, 6)
    '''
    p=[]
    for i in range(m):
        y =[]
        y.append(5)
        y.append(6)
        p.append(y)
    return np.array(p)
'''任取m个点
'''
m = 50
X = np.random.random(size=(m, 2))
P = 0.5*grad_fun(X)
print(np.round(X,3).tolist())
print(np.round(P,3).tolist())

c = np.ones((m, ))
A_ub = np.ones((m, m)) - np.identity(m) * m

'''求解b_ub
'''
```

```
t_ub = []
for i in range(m):
    t = np.sqrt(np.sum(np.square(X[i] - P), axis=-1))
    t[i] *= (1 - m)
    t_ub.append(np.sum(t))
b_ub = np.array(t_ub)
# 求解
res = optimize.linprog(c, A_ub=A_ub, b_ub=b_ub, bounds=[0, None], method='re
print(res)

```